

# MIPS32 Ciclu Unic

## Elemente functionale

- Toate elementele pana la instructiunea stocare functioneaza conform tabelului de trasare si duc la rezultatul corect al instructiunilor executate in asamblare.

## Elemente nefunctionale

- Rezultatul e corect dar nu se salveaza in memorie.

## Probleme intampinate

- Nu am intampinat alte probleme inafara de stocarea in memorie.

## Testare

- Programul a fost testat atat pe placa in cadrul laboratorului cat si in simulator.

## Instructiuni alese suplimentar

### SRA (Shift-Right Arithmetic)

- **Descriere:** Deplasare aritmetica la dreapta pentru un registru, rezultatul fiind memorat in alt registru, repetandu-se valoarea bitului de semn.
- **RTL:**  $\$d \leftarrow \$t \gg h$ ;  $PC \leftarrow PC + 4$
- **Sintaxa:** sra \$d, \$t, h
- **Format:** 000000 00000 ttttt ddddd hhhhh 000011
- **Semnale control:** RegDst $\leftarrow$ 1, ExtOp $\leftarrow$ X, ALUSrc $\leftarrow$ 0, Branch $\leftarrow$ 0, Brgez $\leftarrow$ 0, Jump $\leftarrow$ 0, MemWrite $\leftarrow$ 0, MemtoReg $\leftarrow$ 0, RegWrite $\leftarrow$ 0, AluOp $\leftarrow$ 10, function $\leftarrow$ 000011, AluCtrl $\leftarrow$ 110(>>a)

## SLLV (Shift-Left Logical Variable)

- **Descriere:** Deplasare logica la stanga pentru un registru, cu un numar de pozitii indicat de alt registru, rezultatul fiind memorat intr-un al treilea registru.
- **RTL:**  $\$d \leftarrow \$t \ll \$s; PC \leftarrow PC + 4$
- **Sintaxa:** `sllv $d, $t, $s`
- **Format:** 000000 sssss ttttt ddddd 00000 000100
- **Semnale control:**  $\text{RegDst} \leftarrow 1, \text{ExtOp} \leftarrow X, \text{ALUSrc} \leftarrow 0, \text{Branch} \leftarrow 0, \text{Brgez} \leftarrow 0, \text{Jump} \leftarrow 0, \text{MemWrite} \leftarrow 0, \text{MemtoReg} \leftarrow 0, \text{RegWrite} \leftarrow 0, \text{AluOp} \leftarrow 10, \text{function} \leftarrow 000100, \text{AluCtrl} \leftarrow 111(<<lv)$

## ANDI (AND Immediate)

- **Descriere:** SI logic intre un registru si o valoare imediata, rezultatul ajunge in alt registru.
- **RTL:**  $\$t \leftarrow \$s \& ZE(\text{imm}); PC \leftarrow PC + 4$
- **Sintaxa:** `andi $t, $s, imm`
- **Format:** 001100 sssss ttttt iiiiiiiiiiiiiiiii
- **Semnale control:**  $\text{RegDst} \leftarrow 0, \text{ExtOp} \leftarrow 0, \text{ALUSrc} \leftarrow 1, \text{Branch} \leftarrow 0, \text{Brgez} \leftarrow 0, \text{Jump} \leftarrow 0, \text{MemWrite} \leftarrow 0, \text{MemtoReg} \leftarrow 0, \text{RegWrite} \leftarrow 0, \text{AluOp} \leftarrow 11, \text{function} \leftarrow \text{XXXXXX}, \text{AluCtrl} \leftarrow 001$

## BGEZ (Branch on Greater than or Equal to Zero)

- **Descriere:** Se executa un salt conditionat daca un registru este mai mare sau egal cu 0.
- **RTL:**  $\text{if } \$s \geq 0 \text{ then } PC \leftarrow (PC + 4) + (\text{SE}(\text{offset}) \ll 2) \text{ else } PC \leftarrow PC + 4$
- **Sintaxa:** `bgez $s, offset`
- **Format:** 000001 sssss 00000 oooooooooooooooooo
- **Semnale control:**  $\text{RegDst} \leftarrow X, \text{ExtOp} \leftarrow 1, \text{ALUSrc} \leftarrow 0, \text{Branch} \leftarrow 0, \text{Brgez} \leftarrow 1, \text{Jump} \leftarrow 0, \text{MemWrite} \leftarrow 0, \text{MemtoReg} \leftarrow X, \text{RegWrite} \leftarrow 0, \text{AluOp} \leftarrow 01, \text{function} \leftarrow \text{XXXXXX}, \text{AluCtrl} \leftarrow 100$